Restaurarea imaginilor prin filtrare inversă

Laborator 5, PSS

Table of contents

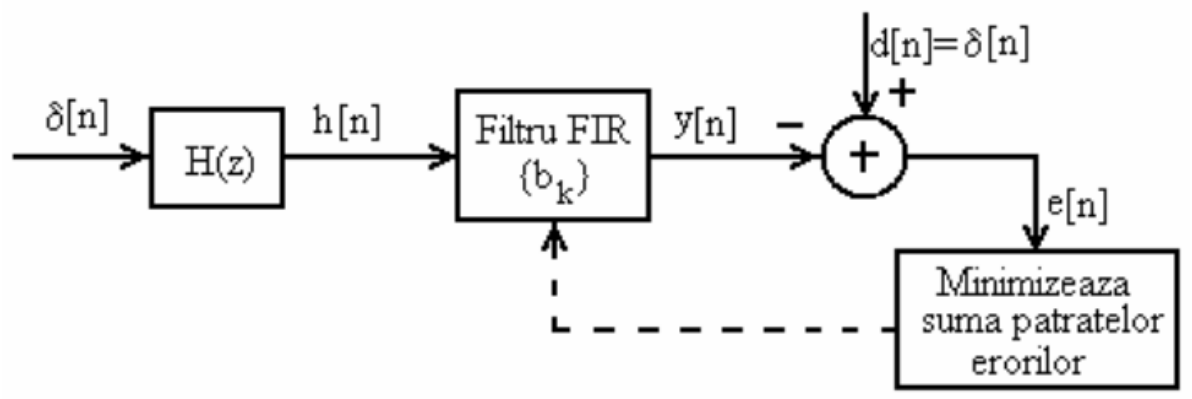
1 Obiectiv

Utilizarea filtrului FIR invers într-o aplicație de procesare de imagini.

2 Noțiuni teoretice

Filtrul **invers** $H_I(z)$ al unui filtru oarecare $H(z)$ este sistemul care anulează efectul lui $H(z)$ asupra unui semnal:

$$H_I\{H\{x[n]\}\} \approx x[n]$$



3 Aplicație practică

Ilustrăm utilizarea filtrului invers prin următoarea aplicație practică.

3.1 Definirea distorsiunilor

Fie cele patru funcții de mai jos, care definesc o serie de distorsiuni asupra unui semnal/imagine de intrare: `distort()`, `distort_more()`, `distort_noisy()`, `distort_delay()`

Cerința 1

Copiați funcțiile de mai jos în fișiere Matlab, pentru a putea fi utilizate ulterior.

```
function y = distort(x)
    L1 = 5;
    coef = [zeros(1,L1) 1.1.^[0:-1:-L1]];
    coef = coef / norm(coef,1);
    coef  = fliplr(coef);  % filter2 expects kernel, not impulse response, it doesn't flip
    coef;
    y = filter2(coef, x, 'same');
end
```

```
function y = distort_more(x)
    L1 = 10;
    coef = [zeros(1,L1) 1.1.^[0:-1:-L1]];
    coef = coef / norm(coef,1);
    coef  = fliplr(coef);  % filter2 expects kernel, not impulse response, it doesn't flip
```

```
    coef;
    y = filter2(coef, x, 'same');
end
```

```
function y = distort_noisy(x)
    L1 = 5;
    coef = [zeros(1,L1) 1.1.^[0:-1:-L1]];
    coef = coef / norm(coef,1);
    coef  = fliplr(coef);  % filter2 expects kernel, not impulse response, it doesn't flip
    coef;
    y = filter2(coef, x, 'same');

    y = y + 0.05*randn(size(y));
end
```

```
function y = distort_delay(x)
    Delay = 10;
    L1 = 5;
    coef = [zeros(1,L1+Delay) 1.1.^[0:-1:-L1]];
    coef = coef / norm(coef,1);
    coef  = fliplr(coef);  % filter2 expects kernel, not impulse response, it doesn't flip
    coef;
    y = filter2(coef, x, 'same');
end
```

3.2 Distorsionarea unei imagini

Cerința 2

Incărcați imaginea `'lena512.bmp'`, convertiți-o la tipul `double`, convertiți-o la `grayscale`, și afișați-o.

Se utilizează funcțiile Matlab:

- `imread()`
- `double()`, urmată de împărțire la 255
- `im2gray()`
- `imshow()`

```
I1 = ...   % original image
I2 = ...   % after preprocessing
...
```

Cerința 3

Distorsionați imaginea apelând funcția de distorsiune `distort()` asupra imaginii, și afișați rezultatul.

Cum arată imaginea distorsionată? Ce tip de distorsiune este aceasta?

```
I3 = ...
imshow(I3)
```

3.3 Refacerea imaginii prin filtrare inversă

Etape:

1. Obțineți răspunsul la impuls, apelând funcția asupra unui semnal de tip impuls unitate
2. Calculați filtrul FIR invers cu funcția din laboratorul trecut
3. Filtrați fiecare linie a imaginii distorsionate cu filtrul invers (filtrare 1-D) și stocați rezultatele într-o nouă imagine.

   Pentru filtrare, utilizați una dintre următoarele două funcții:

   - funcția `filter2(h, I3)`
   - funcția `filter(h, 1, I3(i,:))` pe fiecare linie `i` a imaginii
4. Afișați rezultatul

Cerința 4

Găsiți și afișați răspunsul la impuls al distorsiunii `distort()`

```
h = ...
```

Cerința 5

Calculați filtrul invers cu funcția `FIRinvers()`, afișați coeficienții și răspunsul la impuls.

Cât este $H(z) =$?

```
b =

% Make b horizontal
b = b'

stem(b)
```

Cerința 6

Filtrați fiecare linie a imaginii distorsionate cu filtrul găsit, stocați rezultatele, afișați imaginea finală.

```
...

imshow(Irec)
```

3.4 Cerințe finale

1. Repetați cu alte imagini (`bugs.jpg`, `barbara.png`)
2. Repetați cu celelalte funcții de distorsiune. Când se înrăutățesc rezultatele?